# Estimating the Spatial Extent of Attractors of Iterated Function Systems

D. Canright
Mathematics Dept., Code MA/Ca
Naval Postgraduate School
Monterey, CA 93943

August 1993

**Abstract**

From any given Iterated Function System, a small set of balls that cover the fractal attractor can be simply determined. This gives *a priori* bounds on the region of space in which the attractor may be constructed.

## Introduction

As Barnsley, Demko and others have shown [1, 2, 6, 4], one effective method for producing fractal shapes (in any number of dimensions) is with Iterated Function Systems (IFSs), using the "Chaos Game" algorithm (or some deterministic algorithm). This approach has been used for producing naturalistic shapes [4], finding fractal interpolants to given data [6, p. 274] and fractal approximations of given functions [8], and even for visualizing arbitrary discrete sequences [7]. Indeed, any contractive IFS will give an attractor (usually of fractal dimension); thus it is possible to generate IFSs at random to explore the graphical possibilities, as is done in some educational software [5]. Similarly, because the attractor depends continuously on the parameters in the IFS [1], small data sets from any source could be encoded as IFSs for visualization.

In implementing the IFS method, one important question is the prediction *a priori* of the region of space containing the fractal attractor. Without such a prediction, one could only approximately estimate the spatial extent based on calculating several points of the attractor, with no guarantee that these points are near the bounds. If as a result the portion of space represented in the computation of the attractor is too small, the result will not yield the whole attractor. If the portion of space represented is overly large, then much computational space is wasted, reducing the effective resolution of the computed attractor.

Another concern is when the space itself has natural limits, for example the space of colors in Red-Green-Blue-space representable on a video monitor is limited (more or less) to a unit cube. Hence if an IFS resides in, say, five-dimensional space with two spatial dimensions and three color dimensions, the question becomes whether the attractor (or its projection onto

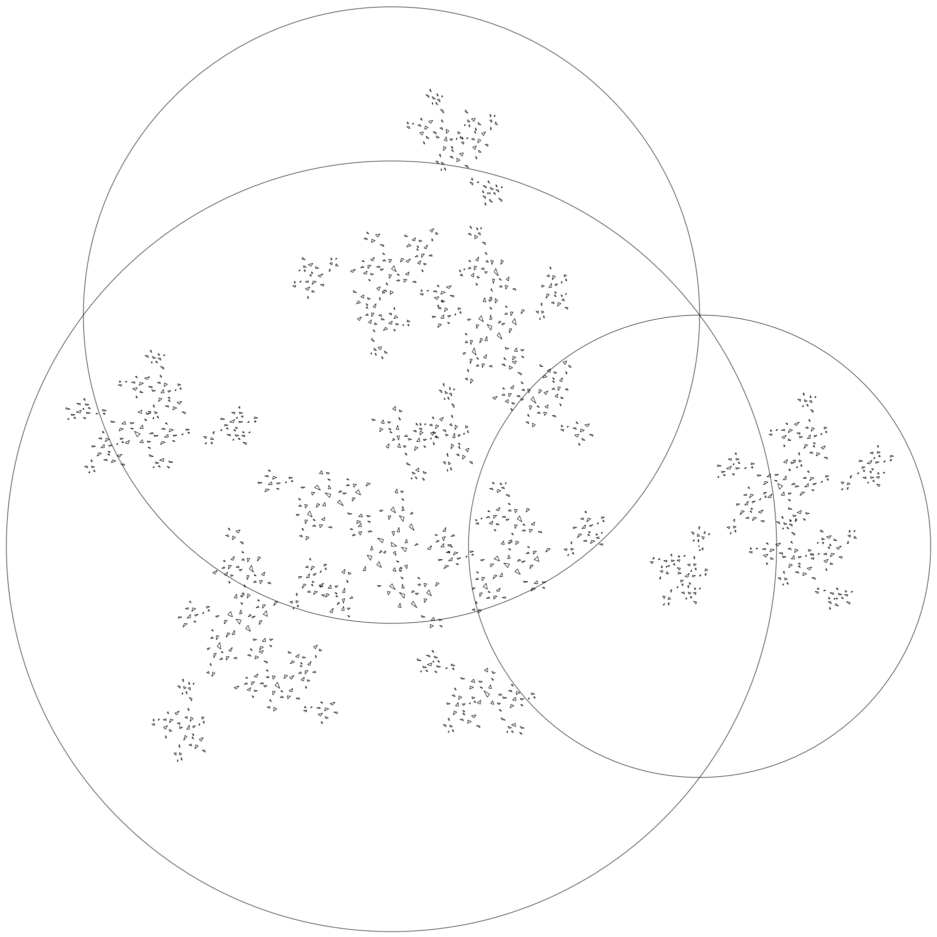

Figure 1: An attractor of an IFS is shown with its envelope of three disks, as computed by the direct algorithm. (This IFS uses affine maps, with $s_1 = \frac{5}{7}$, $s_2 = \frac{1}{3}$, $s_3 = \frac{1}{2}$, $\mathbf{x}_1 = (0,0)$, $\mathbf{x}_2 = (4,0)$, and $\mathbf{x}_3 = (0,3)$.

the limited dimensions) will fit in the space. (At least one implementation [5] includes color in the IFS, though that case uses only a single color dimension, representing hue.)

Here we show how to compute, directly from the IFS, a set of balls whose union contains the attractor as a subset (see Figure 1). The radii of the balls are minimal in a certain restricted sense. This gives reliable bounds on the region of space that must be considered in constructing the fractal. The method is general, independent of the particular space and metric. We first describe the set of balls, then show how to compute their radii and prove that the algorithms work, and lastly give a detailed example.

## 1 IFS Envelopes

An IFS consists of a set of $n$ contraction mappings $w_i : \mathbf{X} \to \mathbf{X}$ on a metric space $\mathbf{X}$ with metric $d : \mathbf{X} \times \mathbf{X} \to \mathcal{R}$. (For the "Chaos Game" algorithm, probabilities $p_i$ are associated with each mapping; this idea has been extended to conditional probabilities [3]. Here only the former case is considered, where the attractor is independent of the non-zero $p_i$.) Assume that for each contraction $w_i$ the contractivity ratio $0 \leq s_i < 1$ and the fixed point $\mathbf{x}_i$ are known, where by definition $s_i$ satisfies $d(w_i(\mathbf{x}), w_i(\mathbf{y})) \leq s_i d(\mathbf{x}, \mathbf{y})$ for all points $\mathbf{x}, \mathbf{y} \in \mathbf{X}$ and $\mathbf{x}_i$ satisfies $\mathbf{x}_i = w_i(\mathbf{x}_i)$. The action of the IFS $W$ on a set $S$ of points in $\mathbf{X}$ is defined as

$$W(S) \equiv \bigcup_{i=1}^{n} w_i(S) \tag{1}$$

where each contraction $w_i$ is applied to the set $S$ in a pointwise sense. The attractor $A$ is the set of points in $\mathbf{X}$ satisfying

$$A = W(A) \quad . \tag{2}$$

That is, the attractor consists of $n$ smaller "copies" of itself.

We seek to cover each of the $n$ "copies" with a closed ball $B_i$ centered on the corresponding fixed point $\mathbf{x}_i$, so the radius $r_i$ must be chosen large enough that $B_i \supset w_i(A)$. Call the union $E$ of these balls the "envelope," in that $E \supset A$ by (2). Then relative to each $\mathbf{x}_i$, every point in the envelope will be within a distance $R_i = \max_j(d_{ij} + r_j)$, where $d_{ij} \equiv d(\mathbf{x}_i, \mathbf{x}_j)$, because for any point $\mathbf{x}$ in $B_j$, $d(\mathbf{x}_i, \mathbf{x}) \leq d_{ij} + d(\mathbf{x}_j, \mathbf{x}) \leq d_{ij} + r_j$. Applying $w_i$ to such a point $\mathbf{x}$ will give an image point $\mathbf{y}$, where $d(\mathbf{x}_i, \mathbf{y}) \leq s_i d(\mathbf{x}_i, \mathbf{x}) \leq s_i R_i$. Hence if the radii $r_i$ are chosen to satisfy

$$r_i = s_i \max_{j \neq i}(d_{ij} + r_j) \tag{3}$$

for $i, j = 1 \ldots n$ then $B_i$ will contain the image $w_i(E)$ of the envelope and so $E$ will contain its own image under the IFS:

$$E \supset W(E) \quad . \tag{4}$$

Iterating the IFS from any starting set ($E$ in particular) yields a sequence of sets that converges to the attractor. Since (4) implies $E \supset W^k(E)$ for any positive integer $k$, the envelope $E$, subject to (3), does indeed contain the attractor.

## 2 Algorithms

But for a given IFS, does one or more sets of radii $r_i$ exist that satisfy (3), and if so, how can they be found? Below we give a very simple, iterative algorithm, as well as a more efficient direct algorithm. The subsequent proof of the validity of the direct algorithm also constructively proves the existence of a solution, and the structure of the algorithm suggests that the solution is unique. (Given that a unique solution exists, the system (3) is equivalent to a linear programming problem with an objective function $f = \sum_{i=1}^{n} r_i$, as pointed out by a reviewer. However, if one wishes to incorporate envelope calculation into an IFS rendering program, the algorithms given here are simpler than including a general simplex method solver.)

### 2.1 Trivial Case: $n = 2$

When $n = 2$ the radii can be determined algebraically. Solving the pair of equations (3) gives:

$$r_1 = \frac{s_1(1 + s_2)}{1 - s_1 s_2} d_{12}$$

$$r_2 = \frac{s_2(1+s_1)}{1-s_1 s_2} d_{12} \quad . \tag{5}$$

But for $n > 2$ there is apparently no closed-form general solution, and the $r_i$ must be found algorithmically.

## 2.2 Simple Iterative Algorithm

A natural approach for $n > 2$ is to start with the pairwise estimates

$$r_{ij} \equiv \frac{s_i(1+s_j)}{1-s_i s_j} d_{ij} \tag{6}$$

$$r_i^{(1)} = \max_{j \neq i} r_{ij} \tag{7}$$

but in most cases the $r_i^{(1)}$ will not satisfy (3). The exceptional case is when $r_{ij} = r_{ik}$ for every $i, j \neq i, k \neq i$, i.e., when for each ball all the pairwise estimates for that ball give the same size. (This case is not always apparent from the attractor; Figure 1 shows such an example.) Otherwise, some of the $r_i^{(1)}$ will be too small to contain some images $w_i(B_j^{(1)})$ of the other balls. Then the obvious iterative scheme to try is

$$r_i^{(k+1)} = \max_{j \neq i} s_i(d_{ij} + r_j^{(k)}) \quad , \quad i, j = 1 \ldots n \quad . \tag{8}$$

Because this approach never overestimates the radii and the iterates are nondecreasing ($r_i^{(k)} \leq r_i^{(k+1)} \leq r_i$), the algorithm must converge. What is not so obvious is that this process always suceeds in at most $n - 1$ iterations, including (7), as shown below. Indeed, there is a direct algorithm (not iterative) that is more efficient when $n$ is large.

## 2.3 Direct Algorithm

The key idea behind the direct algorithm is that the distances $d_{ij}$ can be rescaled to account for the contractivities $s_i$, and the scaled distances $D_{ij}$ can be used to order the contractions $w_i$. Let

$$D_{ij} \equiv \frac{(1+s_i)(1+s_j)}{1-s_i s_j} d_{ij} \quad , \quad i, j = 1 \ldots n \quad . \tag{9}$$

(While $D_{ij}$ is clearly symmetric and non-negative, it is not a metric because it doesn't satisfy the triangle inequality.) Now *reorder* (and relabel) the $w_i$ by decreasing maximum scaled distance, so that

$$i < j \quad \Rightarrow \quad \max_k D_{ik} \geq \max_l D_{jl} \quad , \quad i, j, k, l = 1 \ldots n \quad . \tag{10}$$

In the new order, use the pairwise formula (5) for the first two radii. Then proceed *in order* based on the previous results, letting

$$r_i = s_i \max_{j<i} (d_{ij} + r_j) \quad , \quad i = 3 \ldots n \quad . \tag{11}$$

This is the direct algorithm, which, as shown below, solves (3); an implementation in the C programming language is given in the Appendix.

## 2.4 Proof of Validity

First note that in the exceptional case mentioned above (after 7), all the $D_{ij}$ (for $i \neq j$) are equal. In this case, the direct algorithm will obtain the correct $r_i$ regardless of the order in which they are computed. If not all the $D_{ij}$ are equal, then some of the $r_i$ will need to be larger than the pairwise estimates $r_i^{(1)}$, and hence larger than the $r_{ij}$ in (6), so in general (3) implies

$$r_i \geq \frac{s_i}{1+s_i} D_{ij} \tag{12}$$

for $i, j = 1 \ldots n$.

For the general case, the proof is by induction, showing that each new $r_i$ computed requires no adjustment of those previously computed. Clearly $r_1$, $r_2$ from (5) satisfy (3) for the subset $i, j = 1, 2$. Now for the induction step, assume the first $m-1$ radii, in the order (10), satisfy (3), and hence (12), for $i, j = 1 \ldots m-1$. Choose $r_m$ by (11), and let $k$ be the value of the index $j$ in (11) for which the maximum is achieved. Then by (12) and the ordering (10)

$$r_k \geq \frac{s_k}{1+s_k} D_{km} \quad . \tag{13}$$

Algebraic manipulation of (13) gives

$$r_k \geq s_k[(1+s_m)d_{km} + s_m r_k] = s_k(d_{km} + r_m) \tag{14}$$

so the new $r_m$ requires no alteration of $r_k$.

Similarly, for $i \neq k$, $i < m$

$$\begin{aligned} r_i &\geq \frac{s_i}{1+s_i} D_{im} \\ &> \frac{s_i(1+s_m)}{1-s_m} d_{im} \end{aligned} \tag{15}$$

since $s_i < 1$. Combining (15) with

$$r_i \geq s_i(d_{ik} + r_k) \tag{16}$$

yields

$$\begin{aligned} r_i &> s_i[(1+s_m)d_{im} + s_m(d_{ik} + r_k)] \\ &\geq s_i[d_{im} + s_m(d_{km} + r_k)] = s_i(d_{im} + r_m) \end{aligned} \tag{17}$$

by the triangle inequality. Hence the new $r_m$ requires no adjustment of any of the previous $r_i$ (for $i < m$), and (3) is satisfied for $i, j = 1 \ldots m$; this completes the proof. Note also that because of how the direct algorithm works, the iterative algorithm will compute at least two of the $r_i$ ($r_1, r_2$ in the order (10)) in the initial step, and will find at least one of the other $r_i$ at each successive step, and so can take at most $n-1$ iterations to arrive at the answer.

## 3 Minimality of the $r_i$

Can radii smaller than these $r_i$ be used and still have the $B_i$ cover the attractor? For any particular IFS, the answer is probably yes (as illustrated in Figure 2). The approach given

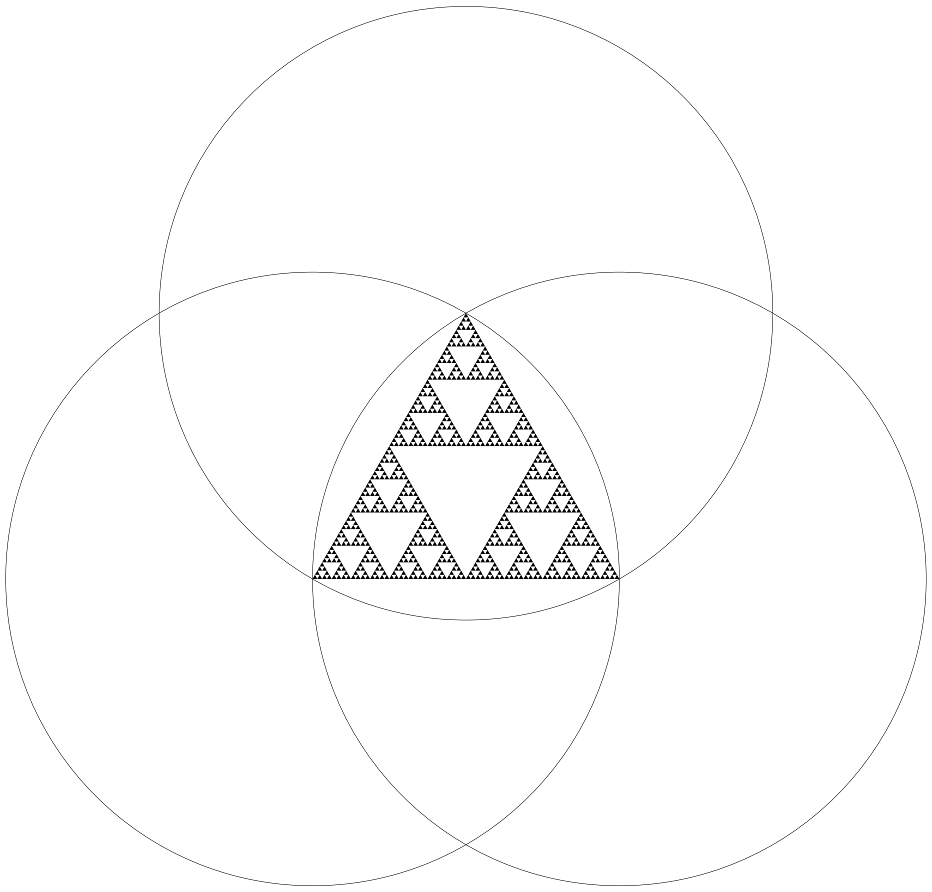

Figure 2: For an equilateral Sierpinski's Triangle (where $w_i(\mathbf{x}) = \frac{1}{2}\mathbf{x} + \frac{1}{2}\mathbf{x}_i$), the $r_i = d_{ij}$ by this method; in this particular case the radii could be half as large.

here uses only a small amount of information about the IFS: the ordering of the $w_i$ based on the maximal scaled distances; the $s_i$; and for each $i$ one determining distance $d_{ik}$ (where the maximum in (11) is achieved). Using more information it may be possible to reduce the size of the $B_i$. But if one considers the set of all the IFSs for which the direct algorithm yields the same $r_i$ in the same way (i.e., same ordered $s_i$ and same $n-1$ determining distances), then the $r_i$ are minimal for that set of attractors (see Figure 3). In fact, one can construct one member of that set such that each image $w_i(A)$ of the attractor includes a point at a distance $r_i$ from the fixed point $x_i$.

To construct this IFS, let $\mathbf{X} = \mathcal{R}$ with the Euclidean metric $d(x,y) = |x-y|$. Let $w_i(x) = -s_i x + (1+s_i)x_i$, and let $x_1 = 0$, $x_2 = d_{12}$, say. Then the attractor includes the extremal points $x_{e1} = x_1 - r_1$, $x_{e2} = x_2 + r_2$, since $x_{e1} = w_1(x_{e2})$ and $x_{e2} = w_2(x_{e1})$. Place each succeding $x_m$ at the determining distance $d_{mk}$ from the determining point $x_k$, in the opposite direction from $x_{ek}$. (Figure 4 illustrates the construction.) Then the attractor will include $x_{em} = x_m \pm r_m = w_m(x_{ek})$. Thus for this one-dimensional attractor $A$, each image $w_i(A)$ will include a point ($x_{ei}$) a distance $r_i$ from $x_i$, so no smaller $r_i$ would suffice.

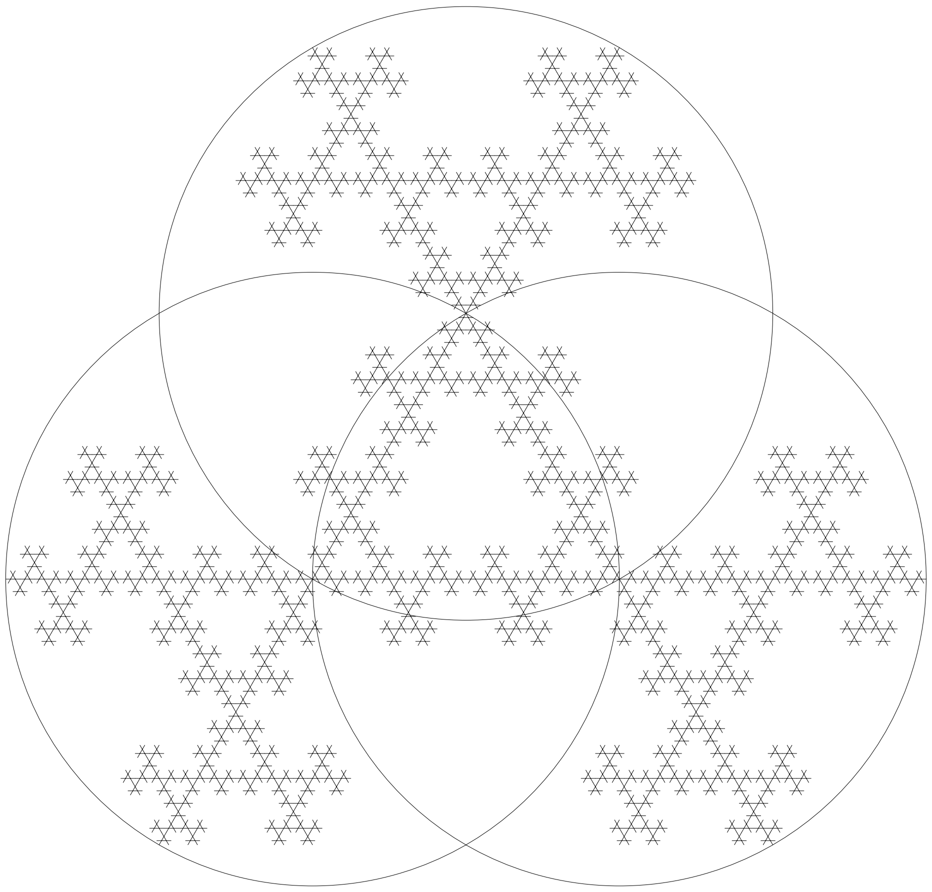

Figure 3: Same $s_i$ and $\mathbf{x}_i$ as the Sierpinski Triangle, but here $w_i(\mathbf{x}) = -\frac{1}{2}\mathbf{x} + \frac{3}{2}\mathbf{x}_i$; in this case the $r_i$ found above are minimal.

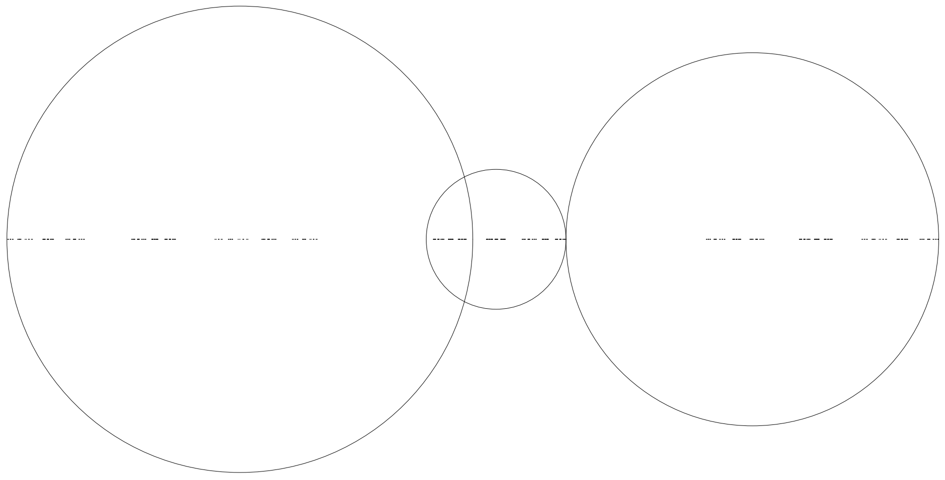

Figure 4: A one-dimensional attractor constructed from the following ordered data: $s_1 = \frac{1}{3}$, $s_2 = \frac{1}{4}$, $s_3 = \frac{1}{7}$, $d_{12} = 1$, and $d_{13} = \frac{1}{2}$. The $r_i$ are minimal for such attractors.

| Map | Scalings | | Rotations | | Translations | |
|---|---|---|---|---|---|---|
| $i$ | $p_i$ | $q_i$ | $\theta_i$ | $\phi_i$ | $e_i$ | $f_i$ |
| 1 | 0 | 0.16 | 0 | 0 | 0 | 0 |
| 2 | 0.85 | 0.85 | -2.5 | -2.5 | 0 | 1.6 |
| 3 | 0.3 | 0.34 | 49 | 49 | 0 | 1.6 |
| 4 | 0.3 | 0.37 | 120 | -50 | 0 | 0.44 |

Table 1: Parameters for Barnsley's fern; all angles are given in degrees.

## 4 Example

For a detailed example of an envelope calculation, consider the now familiar black spleenwort fern fractal of [4]. The IFS for the fern (in two dimensions) consists of affine contractions, each of which has the form

$$w_i\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix} \quad , \tag{18}$$

or more compactly

$$w_i(\mathbf{x}) = M_i\mathbf{x} + \mathbf{b}_i \quad . \tag{19}$$

where $M_i$ is the matrix and $\mathbf{b}_i$ is the offset vector. The various constants are given in [4], but in terms of scaling and rotating each axis, using $p$, $q$, $\theta$, $\phi$, where $a = p\cos\theta$, $b = -q\sin\phi$, $c = p\sin\theta$, $d = q\cos\phi$; Table 1 is adapted from [4, p. 1977].

### 4.1 Finding the $s_i$

The contractivity ratio $s_i$ for an affine map $w_i$ is the largest singular value of the matrix $M_i$. In the first three maps here, both axes rotate together, and so $s_i$ is the larger of $p_i$, $q_i$. In $w_4$, the differential rotation causes a skewing effect, and the singular values of $M_4$ must be found. The simplest way for a real $2 \times 2$ matrix is first to factor out a pure rotation to give a symmetric matrix ($S$), then diagonalize it to find its eigenvalues ($\lambda_1$, $\lambda_2$) as shown below:

$$\alpha = \arctan\left(\frac{c-b}{a+d}\right) \tag{20}$$

$$S \equiv \begin{pmatrix} g & h \\ h & k \end{pmatrix} = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} M \tag{21}$$

$$\beta = \frac{1}{2}\arctan\left(\frac{2h}{g-k}\right) \tag{22}$$

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} = \begin{pmatrix} \cos\beta & \sin\beta \\ -\sin\beta & \cos\beta \end{pmatrix} S \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix} \quad . \tag{23}$$

Then $s = \max(|\lambda_1|, |\lambda_2|)$. This approach also has a nice geometrical interpretation: the effect of multiplying a vector $\mathbf{x}$ by $M$ is to take components of $\mathbf{x}$ in the eigenvector directions, which are at an angle $\beta$ relative to the coordinate axes, scale each component by the corresponding $\lambda$, and rotate the resulting vector by $\alpha$.

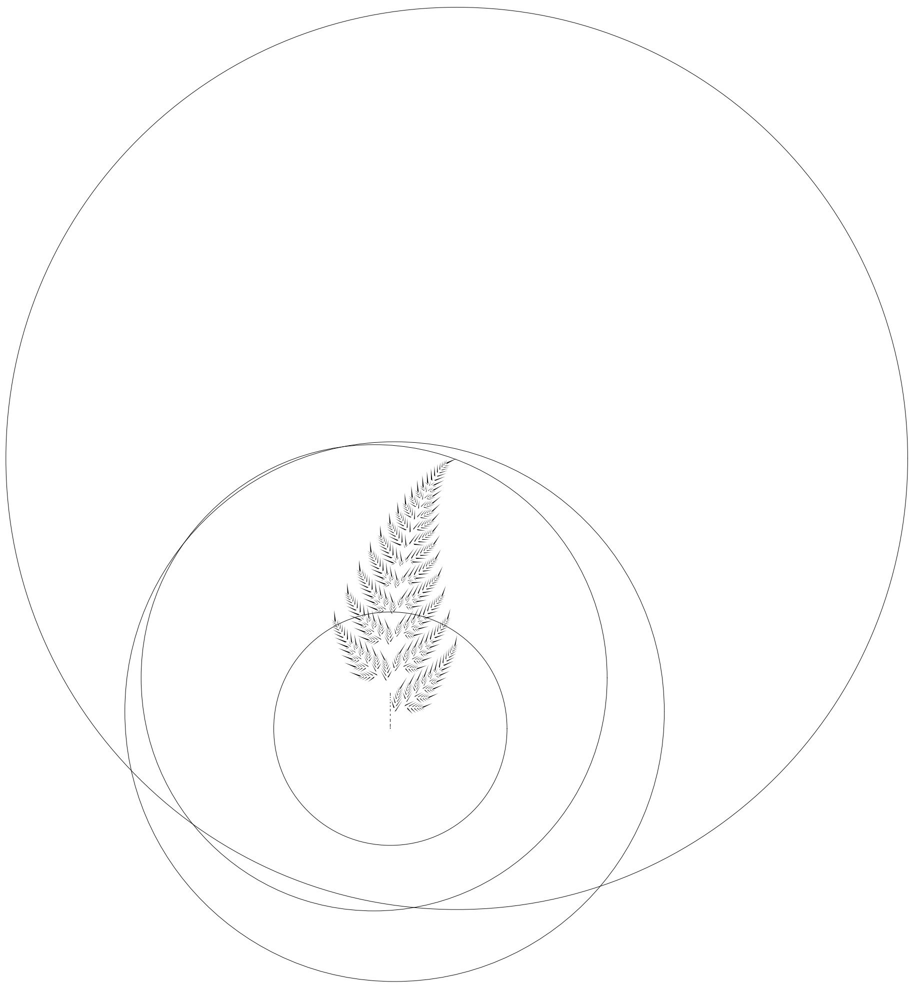

Figure 5: Barnsley's fern [4] and its envelope (see text).

Proceeding as above gives $s_4 = 0.379$. The fixed point $\mathbf{x}_i$ for each map can be found by solving

$$(I - M_i)\,\mathbf{x}_i = \mathbf{b}_i \tag{24}$$

This gives the necessary starting information, summarized as "Input" in Table 2. (For affine maps in higher dimensions, the contractivity ratios are found by singular value decomposition, but for nonlinear maps the ratios and fixed points may be more difficult to find.)

## 4.2 Result

Running the direct algorithm program ("envelope.c") from the Appendix on this input gives the following results:

```
% envelope
Enter number of contraction mappings: 4
Map 1.  Enter scale, x, y: .16 0 0
Map 1: s = 0.160000, x = 0.000000, y = 0.000000
```

| Input | | | | Output | | | |
|---|---|---|---|---|---|---|---|
| $i$ | $s_i$ | $x_i$ | $y_i$ | $D$-order | $D_{max}$ | $d_{det}$ | $r_i$ |
| 1 | 0.16 | 0 | 0 | 4 | $D_{12}$: 25.59 | $d_{12}$: 10.30 | 4.32 |
| 2 | 0.85 | 2.460 | 10.005 | 1 | $D_{24}$: 36.35 | $d_{24}$: 9.65 | 16.70 |
| 3 | 0.34 | -0.602 | 1.884 | 3 | $D_{32}$: 30.26 | $d_{32}$: 8.68 | 8.63 |
| 4 | 0.379 | 0.155 | 0.630 | 2 | $D_{42}$: 36.35 | $d_{42}$: 9.65 | 9.99 |

Table 2: Results of direct algorithm for the fern.

```
Map 2.  Enter scale, x, y: .85 2.45967 10.004734
Map 2: s = 0.850000, x = 2.459670, y = 10.004734
Map 3.  Enter scale, x, y: .34 -0.601889 1.883961
Map 3: s = 0.340000, x = -0.601889, y = 1.883961
Map 4.  Enter scale, x, y: 0.379216 0.155336 0.630251
Map 4: s = 0.379216, x = 0.155336, y = 0.630251
radii in sorted order[orig order](sorted link):
 r1[2](->2): 16.700212
 r2[4](->1): 9.993765
 r3[3](->1): 8.628835
 r4[1](->1): 4.320459
%
```

These results are illustrated in Figure 5, and detailed in Table 2, including the $D$ values used in re-ordering and the determining distances. (While in this example the determining distance for each map derives from the same pair that gives the maximum $D$, that is not always the case.) So if we had no idea how big the fern attractor was, we could use a computational space extending from $x_{min} = x_2 - r_2 = -14.24$ to $x_{max} = x_2 + r_2 = 19.16$ and $y_{min} = y_4 - r_4 = -9.36$ to $y_{max} = y_2 + r_2 = 26.70$ to contain the entire envelope. As it turns out, this is far more space than necessary for the fern itself, but there are many other IFSs, equivalent as far as the direct algorithm is concerned, with much larger attractors (e.g., what if $\theta_2 = \phi_2 = 177.5$ instead).

## 5 Conclusions

To summarize, given any IFS (along with the contractivities and fixed points of each of its constituent contraction mappings), an envelope that covers the attractor can be constructed of one ball for each map, centered on the corresponding fixed point. (In the case of affine maps in two dimensions, an explicit procedure for finding the contractivities and fixed points was given.) The spatial extent of the envelope thus gives a reliable bound on that of the attractor. (In addition, if the balls are disjoint, the attractor is totally disconnected.) Two algorithms were given to calculate the radii of the balls: an iterative one for greatest simplicity, and a direct one for greater efficiency. Both were proven to be effective. While the radii so found may not be minimal for the particular IFS, they are minimal for the set of all IFSs with equivalent information (in the sense described above).

## Acknowlegements

Thanks to Aaron Schusteff and Ismor Fischer for stimulating, informative discussions and helpful suggestions.

## References

[1] M. F. Barnsley, *Fractals Everywhere.* Academic Press/Harcourt Brace Jovanovich, Boston (1988).

[2] M. F. Barnsley and S. Demko, Iterated function systems and the global construction of fractals. *Proc. Roy. Soc. London A* **399**, 243–275 (1985).

[3] M. F. Barnsley, J. H. Elton, and D. P. Hardin, Recurrent iterated function systems. *Constructive Approximation* **5**, 3–31 (1989).

[4] M. F. Barnsley, V. Ervin, D. Hardin, and J. Lancaster, Solution of an inverse problem for fractals and other sets. *Proc. Nat. Acad. Sci. U.S.A.* **83**, 1975–1978 (1986).

[5] D. Canright, FrEd: the Fractal Editor. freeware available as FREDEX.EXE from CompuServe IBMApp forum, Graphics library (1992).

[6] S. Demko, L. Hodges, and B. Naylor, Construction of fractal objects with iterated function systems. *Computer Graphics* **19**, 271–278 (1985).

[7] H. J. Jeffrey, Chaos game visualization of sequences. *Computers and Graphics* **16(1)**, 25–33 (1992).

[8] W. D. Withers, Newton's method for fractal approximation. *Constructive Approximation* **5**, 151–170 (1989).

## Appendix

```
/*
   The following program in C implements the direct algorithm
for determining the "envelope" of an attractor of an Iterated
Function System on R^2, given the contractivities si and the
fixed points (xi,yi).
   written by David Canright, March 1993.
*/
#include <stdio.h>
#include <math.h>

main() {
int npts, i, j, m, n, index[64], link[64];
double d[64][64], x[64], y[64], s[64], r[64], Dmax[64],
   t, tmax, dx, dy;
```

```
char line[81];

/* input n, si, xi, yi */
printf("Enter number of contraction mappings: ");
gets(line); sscanf(line,"%d",&npts);
for (i = 1; i <= npts; i++) {
   printf("Map %d.  Enter scale, x, y: ",i);
   gets(line); sscanf(line,"%lf%lf%lf",s+i,x+i,y+i);
/* enforce 0 <= s < 1 */
   if(s[i]<0.) s[i] = -s[i]; while(s[i]>=1.) s[i] *= 0.1;
   printf("Map %d: s = %f, x = %f, y = %f\n",i,s[i],x[i],y[i]);
   }
/* compute distances dij & maximal scaled Dij */
for (i = 1; i <= npts; i++) Dmax[i] = 0.;
for (i = 1; i <= npts; i++) {
   for (j = i+1; j <= npts; j++) {
      dx = x[i]-x[j]; dy = y[i]-y[j];
      d[i][j] = d[j][i] = t = sqrt(dx*dx+dy*dy);
      t = (1.+s[i])*(1.+s[j])/(1.-s[i]*s[j]) * t;
      if (t > Dmax[i]) Dmax[i] = t;
      if (t > Dmax[j]) Dmax[j] = t;
      }
   }
/* Sort by scaled distances; index points to old order */
index[1] = 1;
for (i = 2; i <= npts; i++) {
   for (m = i; m > 1 && Dmax[i] > Dmax[index[m-1]]; m--)
      index[m] = index[m-1];
   index[m] = i;
   }

/* Direct algorithm; link points to determining distance */
i = index[1]; j = index[2]; link[1] = 2; link[2] = 1;
r[1] = ( s[i]/(1.+s[i]) ) * Dmax[i];
r[2] = ( s[j]/(1.+s[j]) ) * Dmax[j];
for (m=3; m <= npts; m++) {
   i = index[m];
   tmax = 0.;
   for (n = 1; n < m; n++) {
      j = index[n];
      if ( (t = d[i][j] + r[n]) > tmax )
         { tmax = t; link[m] = n; }
      }
   r[m] = s[i] * tmax;
   }
printf("radii in sorted order[orig order](sorted link):\n");
```

```
for (i = 1; i <= npts; i++)
   printf(" r%d[%d](->%d): %f\n",i,index[i],link[i],r[i] );
return(0);
}
```